Many explicit levels

```
macro memoize(f_expr)
  f_name, def = split_longdef(MacroTools.longdef(f_expr))
  d = Dict()
  return :($(esc(f_name))(args…) =
      get!(() -> ($(esc(def)))(args...), $d, args))
end
function split_function_def(ex)
  name = shift!(ex.args[1].args)
  ex.args[1].head = :tuple
  return name, ex
end

@memoize add(a, b) = a + b
```

Many internal levels

- method definition —> macros, metaprogramming
  - code_lowered —> generated functions
    simplified code structure
    - code_typed —> precompiled modules (.ji)
      global inference
      local optimization
      code_warntype – dynamic behavior annotations
      - code_llvm —> external codegen, llvmcall-2.0 Julep
        Intermediate Representation (IR) for low-level optimization
        - code_native —> static system image (.so / .dll / .dylib)
          Machine Code representation